ONTWERPVERSLAG

Evelien van Neutigem, Laura Stolk en Suzanne Wielemaker – Smart Industry 2018/2019

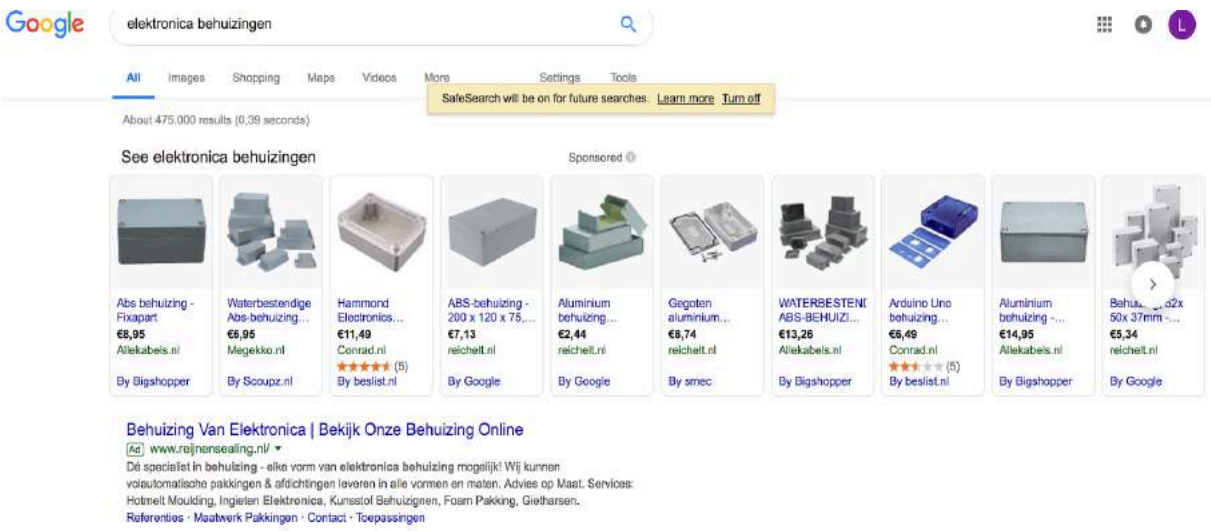
Inleiding

Voor de opleiding Smart Industry is ons als studenten de mogelijkheid geboden om een aantal workshops bij te wonen op het gebied van internet of things, robotica en 3D printing. Het doel van deze workshops is uiteindelijk een applicatie op te leveren die aan het eind van de minor gepresenteerd moet worden. Deze applicatie is tijdens de workshops samengesteld. In dit ontwerpverslag worden de stappen besproken hoe we van niets tot een werkende applicatie zijn gekomen.

Inhoudsopgave

1. Oriëntatie

Zoals je hieronder ziet zijn er veel elektronische behuizingen voor weinig geld zo te koop via internet. Ook zijn er bedrijven zoals:... die behuizing op maat voor je maken.



De meeste behuizingen zijn gemaakt van de stof: Acrylonitrile butadiene styrene. Dit is een soort plastic. De tweede veel voorkomende soort voor een behuizing is aluminium.

De behuizingen zijn in alle vormen en maten, hieronder is een klein overzicht hoe je ze op allerlei sites kunt kopen. Ook kun je specifieke behuizingen kopen, voor bijvoorbeeld je ESP32.

2. Randvoorwaarden

2.1 Plan van Eisen Aquariumcontroller

1. Functionaliteit

1.1. De controller moet semi-permanent de volgende taken uit kunnen voeren:

1.1.2. Watertemperatuur voor de gebruiker op afstand zichtbaar maken.

1.1.3. De gebruiker het filter op afstand aan/uit laten zetten.

1.1.4. De gebruiker de pomp van het luchtsteentje op afstand aan/uit laten zetten.

1.1.5. Gedurende een periode van minimaal 2 weken het aquarium zelfstandig kunnen voeren

1.1.5.1. Hoeveelheid voer en/of periode tussen voertijden op afstand instelbaar door gebruiker

1.1.6 De lamp van het aquarium op tijd aan/uit kunnen schakelen.

1.1.6.1. De aan/uit tijden van de lamp moeten door de gebruiker op afstand kunnen worden ingesteld.

De controller moet betrouwbaar zijn.

1.2.1 De controller moet minimaal 2 weken achtereen correct kunnen functioneren. (zorgvuldig de code testen en debuggen?)

1.2.2 De controller mag niet ontregeld raken door stroomstoringen. Zodra deze weer stroom heeft moet deze weer functioneren zoals voor de stroomstoring begon.

2. Veiligheid en reparatie

Er mogen geen gaten of open kieren zitten in de bovenkant van de behuizing waar makkelijk water in kan.

Wens: Zo ver mogelijk spatwaterdicht als mogelijk om waterschade aan de electronica te voorkomen

De controller (behuizing met electronica) moet na de eerste installatie zonder extra materialen of gereedschappen kunnen worden losgekoppeld en verwijderd van of geplaatst op het aquarium.

Kapotte onderdelen moeten kunnen worden vervangen

2.3.1 De bevestiging van de elektra onderdelen in de behuizing moet kunnen worden losgehaald, dus geen gebruik van permanente verbindingen (lijm o.i.d.)

2.3.2 Alle losse elektronische onderdelen moeten losgehaald kunnen worden van de esp32, bijv door gebruik van stekkertjes in de behuizing. Zo kan alleen het kapotte onderdeel worden vervangen zonder lossolderen.

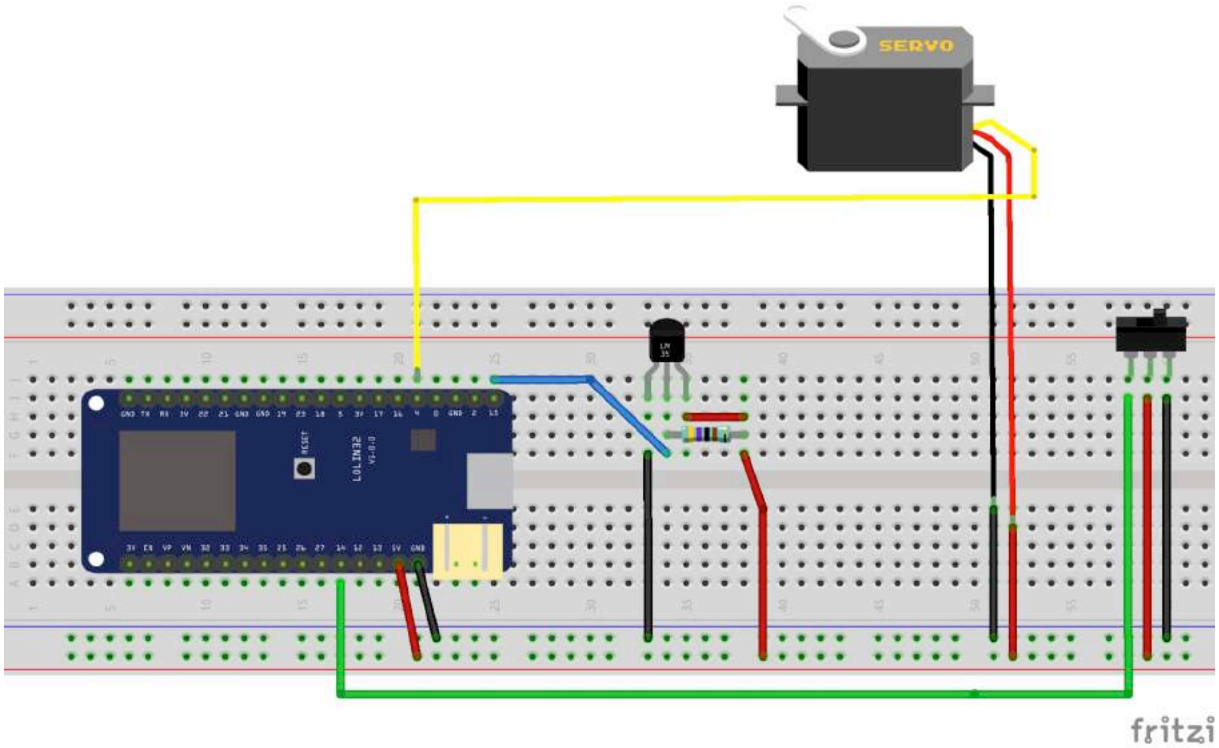
2.2 Onderdelenlijst Aquariumcontroller

3D geprinte parts

1x 3d geprinte behuizing helft A
1x 3d geprinte behuizing helft B
1x 3d geprint kapje (optioneel)
1x 3d geprinte houder voor bevestiging aan het aquarium (optioneel)
1x alle 3d geprinte onderdelen voor een servo voermachine (niet zelf ontworpen, https://www.thingiverse.com/thing:736693)

Inkooponderdelen

1x ESP32 board
1x DS18B20 waterproof dallas temperature sensor
1x Resistor 4,7K
1x XD-FST RF transmmitter 433mhz
3x KlikAanKlikUit remote controlled switch (afstandsbediening niet nodig)
1x TS90D continous rotation servo 9gr
2x Dupont connector 3 draads (M)
3x Dupont connector 3 draads (F)
2x Schroef m3x10

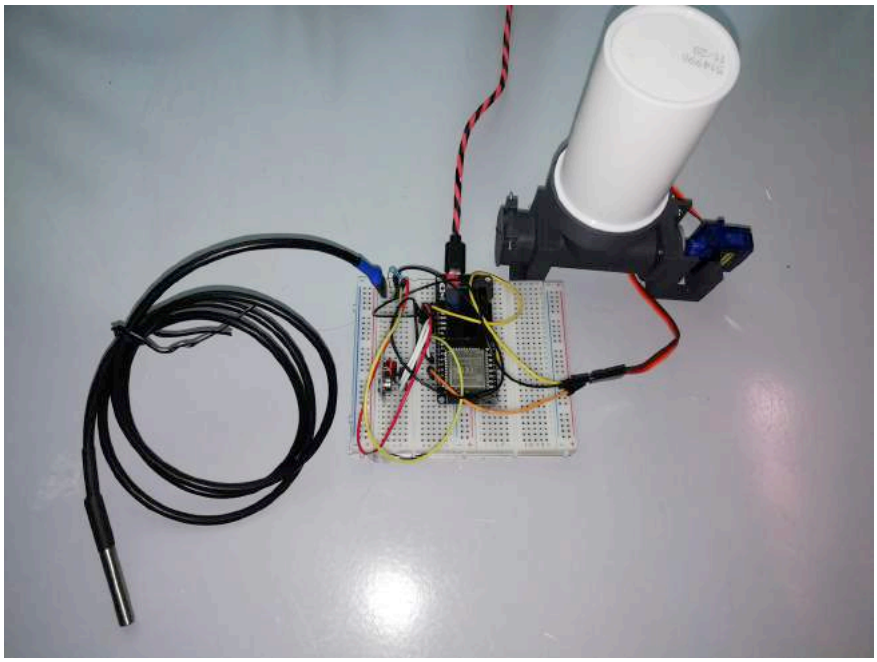


Figuur 1 Schematische weergave van de aquarium controller

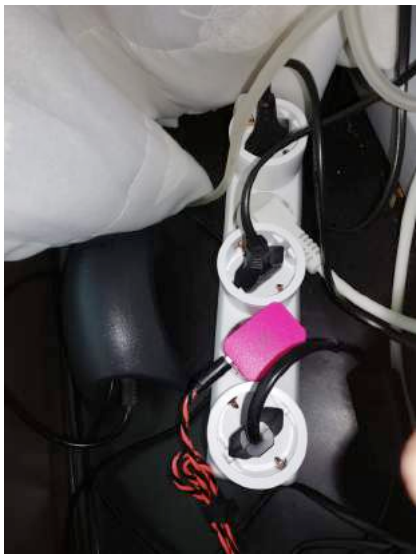
Let op: omdat de Fritzing library niet alle onderdelen bevat, is de ESP32 WEMOS OLED vervangen door een ESP32 WEMOS LOLIN, de DS18B20 door een LM35 en de RF transmitter door een 3-weg schakelaar. Deze zijn aangesloten zoals normaal het missende onderdeel aangesloten zou worden!

2.3 Prototype

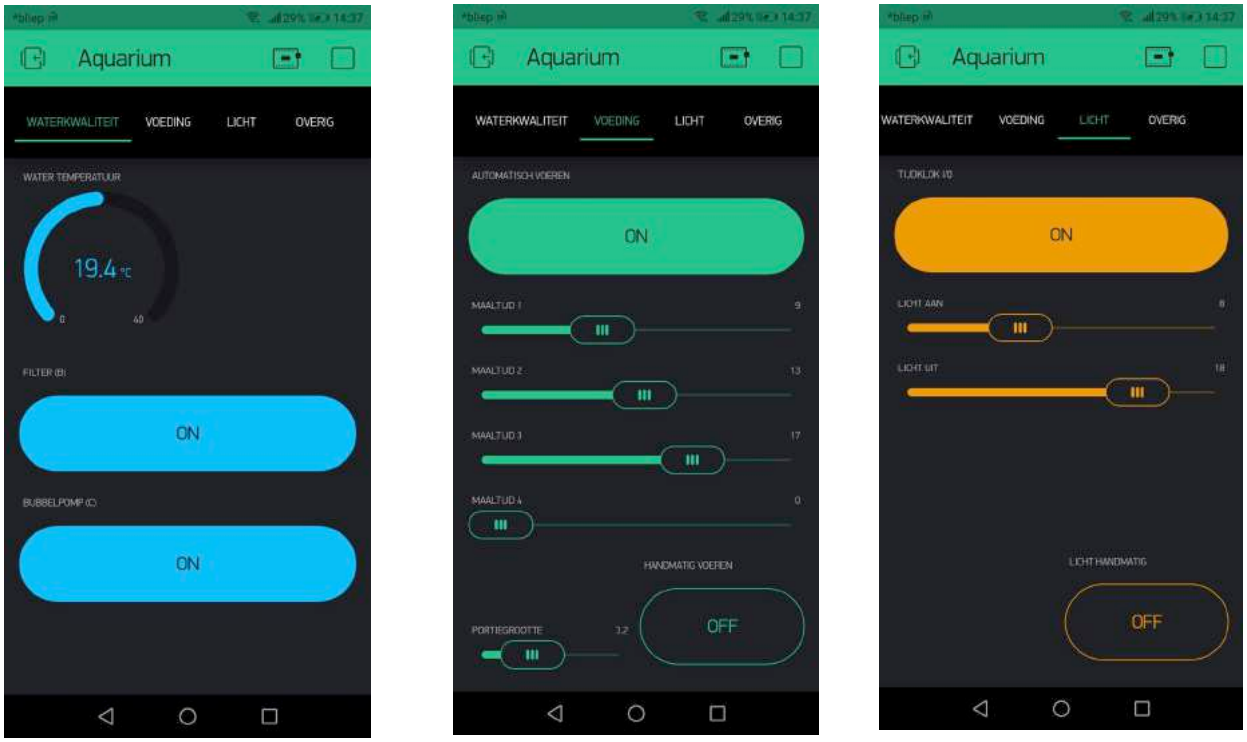
Het prototype is het resultaat van het werken met de ESP32 en kan alle functies uitvoeren die het uiteindelijke product. Het is door de breadboard wel groot, met kwetsbare verbindingen, en door gebrek aan een behuizing zeer gevoelig voor water en vuil! Het is echter wel perfect voor een tijdelijke test om zeker te zijn dat alle bugs uit de code zijn. Dit prototype heeft op het moment van schrijven al meerdere weken succesvol gedraaid. Zelfs bij stroomverlies haalt de ESP32 de laatst ingestelde instellingen uit het geheugen, haalt de tijd van een server en gaat door alsof er niets is gebeurd.



Figuur 2 Het prototype van de controller

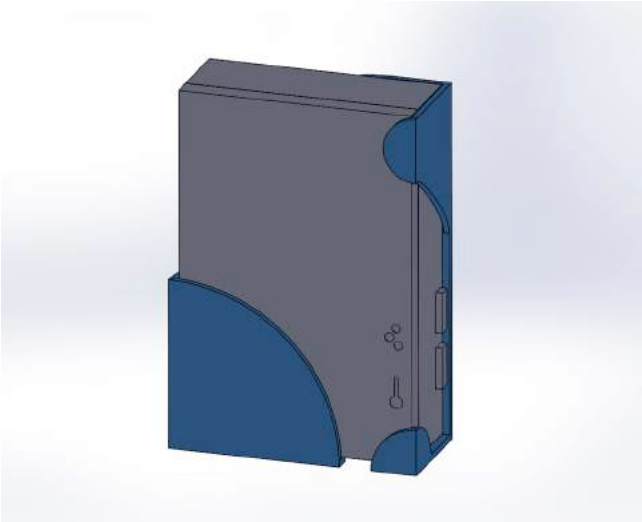


Figuur 3 Stroomvoorziening aquarium controller

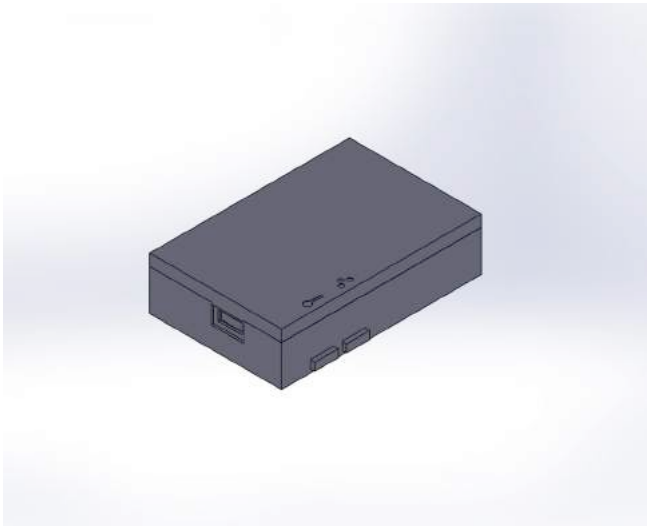


Figuur 4 het prototype van de controller

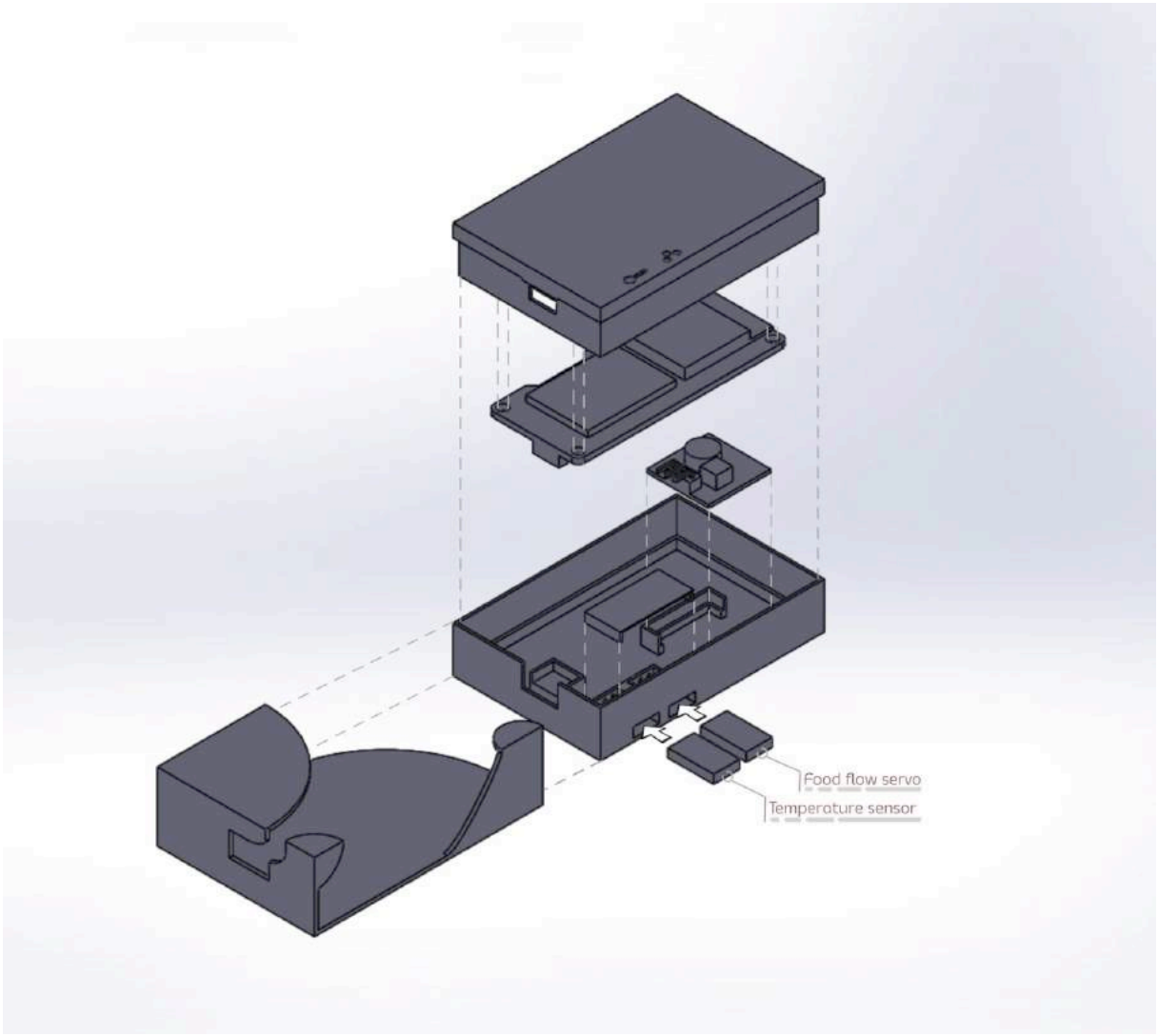
2.4 3D model



Figuur 5 Assembly van de aquariumcontroller in houder

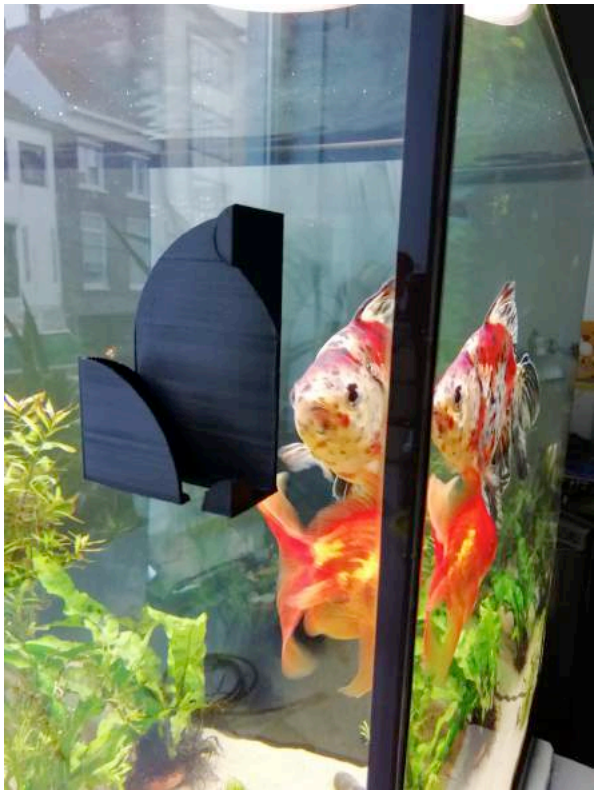


Figuur 6 Assembly van de aquariumcontroller



Figuur 7 Exploded view van de verschillende parts in de assembly

2.5 Product



Figuur 8 Houder gemonteerd aan het aquarium

2.6 Libraries

Switchkaku master (https://github.com/vdwel/switchKaKu)

Blynk (https://github.com/blynkkk/blynk-library)

Onewire (https://github.com/stickbreaker/OneWire)

Dallas (https://github.com/milesburton/Arduino-Temperature-Control-Library)

ESP_32 servo (https://github.com/RoboticsBrno/ESP32-Arduino-Servo-Library/archive/master.zip**)**

Code

```
/* 1x DS18B20 waterproof dallas temp sensor
Red wire -> 5v    black wire -> gnd       yellow wire -> 15   (add a 4,7K pullup resistor between 5v and 15)
 * 1x XD-FST RF transmitter 433mhz
VCC -> 5v        GND -> gnd            ATAD ->  14
 * 1x TS90D continious rotation servo 9g          Red wire -> 5v    brown wire -> gnd orange wire -> 4
 */

 #define BLYNK_PRINT Serial
#define EEPROM_SIZE 64
#define ONE_WIRE_BUS 15
#include "switchKaKu.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <BlynkSimpleEsp32.h>
#include "time.h"
#include <ESP32_Servo.h>
#include "EEPROM.h"

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
Servo myservo;
```

```
#define TRANSMITTERID1 1488126    //Groep 1,2,  (filter)
#define TRANSMITTERID2 1231231    //Groep 1,1 (bubbler)
#define TRANSMITTERID3 3213213    //Groep 1,3  (light)

const int transmitter = 14;
const int servoPin = 4;

char auth[] = "Insert_Blynk_Code_Here" ;
char ssid[] = "Insert_Wifi_SSD_Here";
char pass[] = "Insert_Wifi_Pass_Here";



const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 3600;
//const int   daylightOffset_sec = 3600;        //Summertime
const int   daylightOffset_sec = 0;             //Wintertime

int currentHour;
int lightOn;
int lightOff;
int lightSwitch;
int lightButton;
bool CurrentLightState;
int feedTime1;
int feedTime2;
int feedTime3;
int feedSwitch;
int feedButton;
int lastFed;
float portionSize;
int filterValueSwitch;
int bubblerValueSwitch;

int commit = 0;
int i = 0;
```

```
int addrFilterValueSwitch = 10;
int addrBubblerValueSwitch = 11;
int addrFeedSwitch = 20;
int addrFeedTime1 = 21;
int addrFeedTime2 = 22;
int addrFeedTime3 = 23;
int addrPortionSize = 24;
int addrLightSwitch = 30;
int addrLightButton = 31;
int addrLightOn = 32;
int addrLightOff = 33;




#ifdef __cplusplus                          // Read internal temperature sensor
extern "C" {
#endif
uint8_t temprature_sens_read();
#ifdef __cplusplus
}
#endif
uint8_t temprature_sens_read();



int KakuSwitchState(int a, int b, int c, bool d)     //Checks wether the signal has to be sent or
not, and sends if required. (Transmitter id, group, unit, on/off)
{
  if ( d != CurrentLightState)
  {switchKaku(transmitter, a, b, c, d, 3);}
  CurrentLightState = d;
}

void printLocalTime()
```

```
// Reads the time and prints time and date. Also updates currentHour, which is used for timing the feeder and the light switch
{
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
  currentHour = timeinfo.tm_hour;
}

 void EEPROMUpdate(int addr, int val)
//Checks wether the value to be written to the EEPROM has changed and applies the change if it has
{
  if (val!= EEPROM.read(addr))
  {
  Serial.println("Overwriting....");
  EEPROM.write(addr, val);
  commit = 1;
  }
}
//-------------------------------------------------------
void setup()
{
  Serial.begin(115200);
  Serial.println("start...");

  if (!EEPROM.begin(EEPROM_SIZE))
  {
    Serial.println("failed to initialise EEPROM");
  }
  lightSwitch = EEPROM.read(addrLightSwitch);
  lightButton = EEPROM.read(addrLightButton);
  lightOn = EEPROM.read(addrLightOn);
```

```
 lightOff = EEPROM.read(addrLightOff);
 feedSwitch = EEPROM.read(addrFeedSwitch);
 feedTime1 = EEPROM.read(addrFeedTime1);
 feedTime2 = EEPROM.read(addrFeedTime2);
 feedTime3 = EEPROM.read(addrFeedTime3);
 portionSize = EEPROM.read(addrPortionSize);

 WiFi.begin(ssid, pass);
 while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
 }
 Serial.println(" WIFI CONNECTED");


 Blynk.begin(auth, ssid, pass);
 configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
 myservo.attach(servoPin, 500, 2400);
 sensors.begin();

 delay (10000);

 filterValueSwitch = EEPROM.read(addrFilterValueSwitch);
 Serial.println(filterValueSwitch);
 if (filterValueSwitch == 1)
   {switchKaku(transmitter, TRANSMITTERID1, 1, 2, true, 3);
   Serial.println("filter on..."); }

bubblerValueSwitch = EEPROM.read(addrBubblerValueSwitch);
 Serial.println(bubblerValueSwitch);
 if (bubblerValueSwitch == 1)
   {switchKaku(transmitter, TRANSMITTERID2, 1, 1, true, 3);
   Serial.println("bubbler on..."); }
}
```

```
//--------------------------------------------------
void loop()
{
  Blynk.run();
  printLocalTime();
  sensors.requestTemperatures(); // Send the command to get temperatures

//------------------------------Light control part---------------------------------------
 if (lightSwitch == 1)
  {
    if (currentHour >= lightOn && currentHour <lightOff)
          {
           KakuSwitchState(TRANSMITTERID3, 1, 3, true);
           Serial.println ("Auto: Light ON");
          }
    else  {
           KakuSwitchState(TRANSMITTERID3, 1, 3, false);
           Serial.println("Auto: Light OFF");
          }
          delay (300);
// not sure why this delay is required, but otherwise the serial monitor goes haywire.
Something wrong in the KakuSwitchState(); command?
  }
  else
  {
    if (lightButton == 1)
      {
       KakuSwitchState(TRANSMITTERID3, 1, 3, true);
       Serial.println ("Manual: Light ON");
      }
    else
      {
       KakuSwitchState(TRANSMITTERID3, 1, 3, false);
       Serial.println("Manual: Light OFF");
      }
```

```
      delay (300);
  }
  //------------------------------food control part--------------------------------------
if (feedButton == 1){
  Serial.println ("Manual: feeding");
  myservo.write(0); }

else if (feedSwitch == 1)
    {
     if (currentHour == feedTime1 || currentHour == feedTime2 || currentHour == feedTime3)
     {
       if (lastFed != currentHour)
       {i = 0;}
        if (i < 1){
           myservo.write(0);
           delay(portionSize*100);
           myservo.write(90);
           Serial.println ("Auto: feeding");
           i++;
           lastFed = currentHour;}
       else{
           Serial.println ("Auto: fed"); }
    }
  }
else
   {
   myservo.write(90);
   if(feedSwitch == 1)
     {Serial.println ("Auto: not feeding");}
    else
     {Serial.println ("Manual: not feeding");}
   }


//------------------------------Serial print and eeprom update-------------------------------------
```

```
   Serial.print("Current hour is: ");              Serial.println (currentHour);
   Serial.print("Auto lightning is: ");            Serial.println (lightSwitch);
   Serial.print("Light Start = ");                Serial.println (lightOn);
   Serial.print("Light End = ");                  Serial.println (lightOff);
   Serial.print("Auto feeding is: ");             Serial.println (feedSwitch);
   Serial.print("feedTime: ");                    Serial.println (feedTime1);
   Serial.print("Portionsize is: ");              Serial.println (portionSize);
   Serial.print("The chip temp is ");             Serial.println ((temprature_sens_read() - 32) /
1.8);
   Serial.print("And the water temperature is "); Serial.println
(sensors.getTempCByIndex(0));
   Serial.println ("");

   EEPROMUpdate(addrLightSwitch, lightSwitch);
   EEPROMUpdate(addrLightButton, lightButton);
   EEPROMUpdate(addrLightOn, lightOn);
   EEPROMUpdate(addrLightOff, lightOff);
   EEPROMUpdate(addrFilterValueSwitch, filterValueSwitch);
   EEPROMUpdate(addrBubblerValueSwitch, bubblerValueSwitch);
   EEPROMUpdate(addrFeedSwitch, feedSwitch);
   EEPROMUpdate(addrFeedTime1, feedTime1);
   EEPROMUpdate(addrFeedTime2, feedTime2);
   EEPROMUpdate(addrFeedTime3, feedTime3);
   EEPROMUpdate(addrPortionSize, portionSize);
   if (commit == 1)
   {
   EEPROM.commit();
   commit = 0;
   Serial.println ("Writing changes to EEPROM...");
   }
//-------------------------------------------------------------------------------------------
   delay(1000);
}

//------------------------------Communication with Blynk app-----------------------------------
```

```
 BLYNK_WRITE(V1){
  int pinValueSwitch = param.asInt();
  if (pinValueSwitch ==1) {
     switchKaku(transmitter, TRANSMITTERID1, 1, 2, true, 3);
     filterValueSwitch = 1;}
  else {
     switchKaku(transmitter, TRANSMITTERID1, 1, 2, false, 3);
     filterValueSwitch = 0;}
}

BLYNK_WRITE(V2){
  int pinValueSwitch = param.asInt();
  if (pinValueSwitch ==1) {
      switchKaku(transmitter, TRANSMITTERID1, 1, 2, false, 6);  }
}

BLYNK_WRITE(V3){
  int pinValueSwitch = param.asInt();
  if (pinValueSwitch == 1) {
     switchKaku(transmitter, TRANSMITTERID2, 1, 1, true, 3);
     bubblerValueSwitch = 1;
     }
  else {
     switchKaku(transmitter, TRANSMITTERID2, 1, 1, false, 3);
     bubblerValueSwitch =0;
     }
}

BLYNK_WRITE(V4){
  int pinValueSwitch = param.asInt();
  if (pinValueSwitch ==1) {
      switchKaku(transmitter, TRANSMITTERID2, 1, 1, false, 6);  }
}
```

```
BLYNK_WRITE(V15){
  int pinValueSwitch = param.asInt();
  if (pinValueSwitch ==1) {
      switchKaku(transmitter, TRANSMITTERID3, 1, 3, false, 6);  }
}
BLYNK_READ(V5)
{
  sensors.requestTemperatures();
  Blynk.virtualWrite(V5,sensors.getTempCByIndex(0));
}
BLYNK_READ(V16){Blynk.virtualWrite(V16,((temprature_sens_read() - 32) / 1.8));}

BLYNK_WRITE(V14){lightButton = param.asInt();}
BLYNK_WRITE(V6){lightSwitch = param.asInt();}
BLYNK_WRITE(V7){lightOn = param.asInt();}
BLYNK_WRITE(V8){lightOff = param.asInt();}

BLYNK_WRITE(V9){feedButton = param.asInt();}
BLYNK_WRITE(V10){feedSwitch = param.asInt();}
BLYNK_WRITE(V11){feedTime1 = param.asInt();}
BLYNK_WRITE(V12){feedTime2 = param.asInt();}
BLYNK_WRITE(V13){feedTime3 = param.asInt();}
BLYNK_WRITE(V17){portionSize = param.asFloat();}
```

3. Ontwerpproces

Aan de hand van een mini-bouwpakketje zijn we in Arduino aan de slag gegaan om een werkende internet of things toepassing te maken. Na het opdoen van enige basiskennis op het gebied van internet of things. Was het goed duidelijk wat er moest gebeuren. Waar we wel tegenaan liepen was dat de handleiding soms niet helemaal duidelijk was, soms doordat er ICT-jargon is gebruikt en soms omdat het niet heel specifiek stond beschreven. Uiteindelijk konden we als bedrijfskundige studenten er met hulp van medestudenten en docenten ook een werkende toepassing maken.

Na het instellen van het IoT apparaatje hebben we met behulp van solidworks een case kunnen ontwerpen voor de ESP32. Omdat iedereen een plant laat zien tijdens de seminars en Suzanne zelf ook met een 3D-print project bezig was hebben we uiteindelijk besloten om de case van de ESP32 van Suzanne te gebruiken. Hierdoor kunnen we laten zien dat er meerdere toepassingen zijn met een vergelijkbaar systeem. Om toch wijzer te worden van het ontwerpproces wat Suzanne heeft gevolgd hebben we regelmatig kunnen meekijken. Daarnaast hebben we het volgende interview afgenomen:

Waarom gebruik je solidworks? Er zijn ook vele andere programmas die makkelijker zijn namelijk.
Deels macht der gewoonte; ik heb vanuit school geleerd te werken met Solidworks. Maar juist wat Solidworks moeilijk maakt, namelijk dat het zo uitgebreid is, maakt t super om mee te werken. Het parametrisch ontwerpen (ontwerpen door alles met elkaar te verbinden; hetzij door maten hetzij door relaties, bijv gat 1 staat op middelpunt van lijn 4) vind ik heel fijn, omdat je op elk moment 'terug de tijd' in kan gaan, daar een schets of functie aanpassen, weer vooruit kan rollen en alles past automatisch aan.

Hoe moet je alles meten om te weten of het formaat etc goed is?
Daar heeft men een schuifmaat voor uitgevonden, he! Ik heb zelf een 'virtual twin' gemodelleerd van de onderdelen die ik in de casing wil hebben, met de belangrijkste functionele maten, bijv de maten van het bordje, de afstand tussen de gaatjes, locatie van de usb poort en de dikte van belangrijke chips. Deze heb ik in een Solidworks assembly gezet zodat ik meteen kan zien of de maten van mijn casing ongeveer kloppen.

Hoe stuur je het vanuit Solidworks naar de printer?
In Solidworks het model opslaan als een .stl. Dit model laad ik in een slicer, in mijn geval Cura. Deze maakt er een 3d printbaar model van: dus deze zet er supports in, en maakt een solide model hol met infill, en zet dit model om in een 'route' die de printkop volgt. Deze 'route' is een .gcode bestand. Deze kun je via je laptop naar je printer laden of de printer van een sd kaart laten lezen, maar ik upload naar een 3d print server (raspberry pi met octoprint OS) die de code dus naar de printer stuurt.

Welke printtechniek is het beste voor het aquarium?
Een aquarium kun je maar beter kopen en niet printen... Maar als je de aquariumcontroller casing bedoeld, dat is een lastige vraag. FFF/FDM is moeilijk echt waterdicht te maken dus waarschijnlijk zou iets van polyjet of SLA mooier en beter waterdicht zijn. Alleen is daarvoor het prijskaartje algauw weer te groot, en ik zelf heb alleen toegang tot FDM printers of inktjet. Inktjet is erg duur en bros materiaal en daarmee ongeschikt; daarom heb ik toch gekozen toch mijn casing te optimaliseren voor FDM.

Waarom zou je een chassis maken voor het (de) aquarium (controller)?*
Om 2 redenen:
-Estetische redenen. Niet iedereen vind het mooi om de draden en componenten te kunnen zien.
-Functioneel. Waar een aquarium verschoond word vallen spetters, en electronica en water gaan slecht samen! Daarnaast worden op deze manier meerdere componenten 1 product, wat dan weer

beter is voor het meenemen van de controller en de levensduur van het product (de verschillende componenten bewegen niet meer ten opzichte van elkaar)

Hoe ben je aan het ontwerp van je aquarium gekomen?
Origineel had ik het probleem dat ik in vakanties en weekenden weg ben, maar mijn vissen moeten toch gewoon 2 of 3x per dag hun brokjes krijgen omdat ze anders de echte planten gaan aanvreten en ik dan na elke vakantie nieuwe planten moet kopen. Origineel was dat een heel simpel arduinootje met 2 servo's (https://www.thingiverse.com/thing:312572) maar dit was lastig met instellen, onbetrouwbaar bij stroomstoringen (waarbij dus door een korte stroomdip de boel zo ontregeld raakt dat de vissen meerdere dagen zonder eten zitten) en niet mooi afgewerkt. Later zag ik dat iemand op het internet een IoT oplossing had gemaakt voor het regelen van het aquarium (https://www.thingiverse.com/thing:2137857) ook met blynk, en dat is een beetje mijn inspiratie geweest. Alleen vind ik deze uitwerking niet mooi; de behuizing met geintegreerd voersysteem is eigenlijk ronduit lelijk, moet altijd op dezelfde plek in zicht gemonteerd zijn en is voor geen meter spatwaterdicht.

Ook op electronicaniveau heb ik ook heel wat veranderd; zo gebruik ik de lampen die in mijn aquarium ingebouwd zijn en stuur ik ze slechts aan. De 'bestaande' IoT oplossing gebruikte een RGB spotje, wat heel leuk is omdat je naast aan/uit ook de kleur en intensiteit van het licht kan veranderen maar niet functioneel. Planten groeien het best bij bepaalde kleuren licht, waar deze aquariumlampen dus ook speciaal voor ontwikkeld worden; en ik vermoed dat vissen ook niet heel blij worden van een discolicht in de bak ;)

Welke stappen heb je doorlopen in het proces?
(nog voor de minor) orienteren; er zijn vergelijkbare systemen gemaakt door mensen op bijv. thingiverse
-Inventariseren; wat moet ik wekelijks allemaal doen en hoe veel? -> op papier gekrabbeld
-Digitaliseren; welke handelingen kan ik door sensoren laten doen en hoe? Heel belangrijk, hoe kan ik zorgen dat een stroomstoring niet het hele systeem in de war gooit? Welke sensoren heb ik nodig? Wat kost dat? -> grotendeels bijgekrabbeld op papiertje van stap 1
-Bestellen/printen van de eerste onderdelen
-Schrijven programma voor de esp
-Testen; eerst een paar weken 'droog', zodra hij betrouwbaar genoeg leek heb ik m chassisloos op het aquarium gezet -> kan zo wel even een foto maken
-Onderdelen opmeten en modelleren in Solidworks
-(Op papier) globaal design voor het chassis gemaakt, nadenkend over bijv de locatie van stekkeraansluitingen (usb aansluiting naar beneden gericht om te voorkomen dat er water in valt), zo waterdicht mogelijk maken etc.
-Chassis functioneel modelleren in Solidworks, in deze fase worden ook de laatste tweaks toegevoegd. De schets is niet identiek aan het uiteindelijke model.
-Eerste testprint en testen; passen de parts erin zoals ik zou willen? Gaat alles dicht? Kloppen de toleranties? -> Nee, de esp past er helaas NET niet in
-Aanpassen model en laatste decoraties toevoegen. <-momenteel mee bezig
-Nieuwe print; werkt ie, dan ga ik de hardware solderen en installeren, werkt ie niet word t model weer aangepast natuurlijk

Hoe lang duurt het om een chassin te maken?
Oef... Ik ben begonnen met op papier schetsen na de laatste arduino les, dat was maar n uurtje ofzo. Het modelleren ben ik mee begonnen na 25-11 en heb ik gemiddeld ongeveer 2 a 3 uur per week aan besteed (denkik). Dan nog een testprint van 1,5 uur en een final print van 1, uur dus reken een uurtje of 12? Dit is wel de eerste keer dat ik een vergelijkbaar model maak, dus tis een beetje stoeien nog

Is het chassin ook na 1x precies goed? Of kom je altijd nog achter dingetjes?*
Nee zeker niet. Op elke stap in je ontwerpproces kom je wel weer dingen tegen die toch beter anders hadden gekund, om estetische of functionele redenen. Omdat ik niet echt een tijdslimiet had heb ik dit steeds weer in mijn ontwerp doorgevoerd, of het nou cruciaal was of niet.

Wat is jou ervaring met 3D printen?
Ik heb nu ongeveer een jaar een FDM printertje, mijn vriend heeft een betere sinds ongeveer een half jaar denkik? Dus ik ken de basics van het modelleren voor deze printtechniek, weet hoe je met een slicer werkt, hoe je een printer calibreert etc. Helaas heb ik zelf nog redelijk slechte resultaten, kan aan slecht filament liggen, slechte printer of gewoon slechte settings, maar ik ben er nog niet achter welke van de 3 het is... Gelukkig heeft mn vriend betere resultaten haha

Heb je al meer geprint etc.
Zeker. Er zijn alleen al zoveel leuke modellen te vinden op het internet, je hoeft niet eens zelf te kunnen modelleren om gave dingen te printen!

Wat kun je ons meegeven over het 3D printen?
Ironisch genoeg is dat nog wel het minst interessante deel van het ontwerpproces... Aanzetten en printen maar.